Multi-client Predicate-Only Encryption for Conjunctive Equality Tests

Tim van de Kamp[1](✉), Andreas Peter[1], Maarten H. Everts[1,2], and Willem Jonker[1]

[1] University of Twente, Enschede, The Netherlands
{t.r.vandekamp,a.peter,maarten.everts,w.jonker}@utwente.nl
[2] TNO, Groningen, The Netherlands

Abstract. We propose the first multi-client predicate-only encryption scheme capable of efficiently testing the equality of two encrypted vectors. Our construction can be used for the privacy-preserving monitoring of relations among multiple clients. Since both the clients' data and the predicates are encrypted, our system is suitable for situations in which this information is considered sensitive. We prove our construction plaintext and predicate private in the generic bilinear group model using random oracles, and secure under chosen-plaintext attack with unbounded corruptions under the symmetric external Diffie–Hellman assumption. Additionally, we provide a proof-of-concept implementation that is capable of evaluating one thousand predicates defined over the inputs of ten clients in less than a minute on commodity hardware.

Keywords: Multi-client functional encryption
Predicate-only encryption · Privacy-preserving multi-client monitoring

1 Introduction

Predicate encryption (PE) [17] is a special type of encryption that supports the evaluation of functions on encrypted data. On a conceptual level, in predicate encryption a ciphertext of a message m is associated with a descriptive value x and a decryption key SK_f with a predicate f. The decryption of a ciphertext using a key SK_f only succeeds if the predicate $f(x)$ evaluates to TRUE. Special-purpose variants of this notion include identity-based encryption (IBE) [3], attributebased encryption (ABE) [28], and hidden vector encryption (HVE) [6]. Another variant of PE is *predicate-only encryption* [17,30]. In predicate-only encryption, ciphertexts do not contain a message m, but merely consist of an encryption of the descriptive value x. In this case, the decryption algorithm returns the outcome of the predicate f evaluated on the predicate subject x, that is, $f(x)$.

The concept of PE can be generalized to functional encryption (FE) [5,25], in which the decryption of a ciphertext using a key SK_f for a (not necessarily predicate) function f does not return the original plaintext m, but the

S. Capkun and S. S. M. Chow (Eds.): CANS 2017, LNCS 11261, pp. 135–157, 2018.
https://doi.org/10.1007/978-3-030-02641-7_7

value $f(m)$ instead. More recently, Goldwasser et al. [15] formally defined multiclient functional encryption (MC-FE). MC-FE is a type of secret key encryption in which n distinct clients can individually encrypt a message m_i using their secret encryption key usk_i. Using a decryption key for an n-ary function f, the decryption algorithm takes as input the n ciphertexts of the clients and returns $f(m_1, \ldots, m_n)$. Although FE for generalized functionalities [14,15] is an active field of research and of great theoretical interest, FE constructions for a restricted family of functions (such as predicates) are often far more efficient than FE schemes for arbitrary polynomially sized circuits. For example, most works in the area of MC-FE for generalized functionalities rely on inefficient primitives such as indistinguishability obfuscation or multilinear maps.

In this work, we propose the first *multi-client* predicate-only encryption scheme. Our construction can evaluate an n-ary predicate f on the descriptive values x_i coming from n distinct clients. The type of predicates that we can evaluate using our construction is restricted to conjunctive equality tests. To put it simply, our multi-client predicate-only encryption (MC-PoE) scheme is capable of testing the equality of two encrypted vectors. One of these vectors is determined by the decryption key, while the other vector is composed of ciphertexts from several distinct clients. We also provide an extension to our construction in which the decryption keys may contain wildcard components. A wildcard component in the decryption key indicates that it does not matter what the client corresponding to that vector component encrypts: any value matches the wildcard. An attentive reader familiar with the concept of HVE [6] will recognize the functional similarity between the two concepts. However, a crucial difference in our construction is that the ciphertext vector is composed of the ciphertexts from multiple clients, instead of being generated by a single party. A further comparison of related work is discussed in Sect. 1.2.

Our multi-client predicate-only encryption construction uses pairing-based cryptography and satisfies two distinct security notions. The first notion encompasses both the *attribute-hiding* [17] (also termed *plaintext-privacy* [30]) and *predicate-privacy* [30] properties of predicate encryption. Informally, these properties guarantee that an adversary can neither learn the value x of a ciphertext, nor learn the predicate from a given decryption key. Since we construct a multiclient scheme, we choose to adapt the established MC-FE security requirement [15] for our *full security* notion of multi-client predicate-only encryption. This full security notion protects against an attacker that has oracle access to both the key generation algorithm and the encryption algorithm. In the associated security game, the adversary is additionally allowed to statically corrupt clients. We prove our construction secure in the generic bilinear group model using random oracles. We also propose the (intuitively weaker) *chosen-plaintext security* notion, in which an attacker has only oracle access to the encryption algorithm, but can instead corrupt an *unbounded* number of clients. We prove our construction secure under this second notion in the standard model using the symmetric external Diffie–Hellman (SXDH) assumption.

Our construction is designed to be simple and fast. We have implemented and analyzed our construction to evaluate whether it is efficient enough to run in practice. In our proof-of-concept implementation, clients can encrypt their values in about 2.6 ms, while decryption keys, depending on the number of vector components, can be created in less than a second. The Test algorithm, used to evaluate the predicate on the multiple inputs, scales linearly in the number of inputs and requires only 0.10 s for the comparison of vectors of length 20.

1.1 Motivating Use Cases

Privacy-preserving monitoring over encrypted data is one of the main applications for multi-client predicate-only encryption. For example, consider the monitoring of a system comprised of various independent subsystems. We want to raise an alarm when a dangerous combination of events at the various subsystems occurs. By centrally collecting status messages of the individual systems, we can check for such situations. Such a central collection of status messages additionally avoids the need for costly interactions between the various systems. However, if these status messages are considered sensitive, the monitoring cannot be done on the cleartext messages. Multi-client predicate-only encryption overcomes this problem by allowing a monitor to evaluate an n-ary predicate over multiple ciphertexts and raise an alarm when the predicate returns TRUE.

A careful reader might realize that encryption of the status messages is not a sufficient requirement. If the monitor can check arbitrary predicates, it can as well recover the individual plaintext status messages[1], making its encryption useless. Therefore, we have to require that another party issues the decryption keys to the monitor. Since we can consider the monitor to be a third party, it is unlikely that it is allowed to learn the predicates, making a strong case for the requirement of both plaintext privacy and predicate privacy.

The functionality of our construction is developed with the applications in the critical infrastructure (CI) domain in mind. The benefits of information sharing are widely acknowledged [27], but stakeholders still very reluctant in sharing their information with other parties [12,23,33]. We give two concrete use cases.

- *Detection of coordinated attacks.* While a single failure of a system in CI may occur occasionally, a sudden failure of multiple systems from distinct CI operators, could be an indication of a large scale cyberattack. By centrally monitoring the “failure”/“running” status messages of the CI operators, a warning can be given to the national computer emergency response team whenever a combination of systems fails, allowing further investigation of the failures. Additionally, instead of sharing just binary messages to indicate whether a system has failed, it is also helpful to share and monitor *cyberalert levels*. These cyberalert levels from different clients are used to get an improved situational overview [20].

---

[1] For example, the monitor could create a decryption key for a predicate evaluation of a single message, e.g., $f(x_1, \ldots, x_n) = \text{TRUE}$ if and only if $x_1 = 0$.

- *Monitoring of dependencies among* CI *operators.* There exist many dependencies among various CIs [21], making it possible for disruptions to easily propagate from one infrastructure to another [11]. By timely reporting status messages on supply, a central authority can determine whether supply will meet demand and otherwise instruct parties to prepare their backup resources. Similarly, the sharing of compliance status (e.g., whether they can be met or not) can be used to take the right security measures at another party [20].

1.2 Related Work

A multi-*input* functional encryption (MI-FE) [15] scheme is FE scheme that supports the computation of functions over multiple encrypted inputs. Examples of special-purpose MI-FE include property-preserving encryption [26], such as for ordering [4,10] or equality [35], and multi-input inner product encryption (MI-IPE) [2]. The MI-IPE scheme by Abdalla et al. [2] is capable of computing the inner product of two vectors, i.e.,the decryption algorithm returns a scalar. This should not be confused with an inner-product *predicate* encryption scheme where *predicates* (with a TRUE/FALSE result) can be evaluated by an inner product. A private-key, multi-*client* FE (MC-FE) scheme [15,16] is a variant of MI-FE. There are two key differences between the two notions. Firstly, MC-FE requires that the ciphertexts for the function inputs are generated by individual *distinct* parties, while in MI-FE it is allowed to have only a single encryptor for all the inputs. Secondly, in MC-FE the ciphertexts are associated with a *time-step* [15] or *identifier*. Such an identifier is used to prevent mix-and-match attacks: decryption only works when all ciphertexts are associated with the same identifier.

Although not recognized as such, several special-purpose MC-FE schemes have already been proposed in literature. Shi et al. [31] propose a construction for the privacy-preserving aggregation of time-series data. Their construction allows a central party to compute and learn the sum over encrypted numbers, without learning the individual numbers themselves. Decentralized multi-authority attribute-based encryption (MA-ABE) [19] can also be considered a form of MC-FE. In MA-ABE, several decryption keys, issued by different authorities and associated with an identifier, need to be combined to decrypt a single ciphertext. The similarity becomes apparent once we swap the roles of the ciphertext and decryption keys.

Wildcards have been used in PE before by Abdalla et al. [1] in IBE and by Boneh and Waters [6] in HVE. These works differ from our work in several aspects. Most importantly, our construction is a multi-client variant instead of single-client. If we would apply a single-client construction in a multi-client setting, we would leak the individual predicate results for each party. Secondly, we achieve both plaintext privacy and predicate privacy, which is known to be impossible to accomplish in the public-key setting [30] ([1,6] are in the public-key setting). Finally, we look at predicate-only encryption, not at regular PE in which the ciphertexts may also contain an encrypted *payload* message.

Numerous PE schemes are used for searchable encryption (SE) [7]. However, we see no great benefit in applying MC-PoE as SE scheme. MC-PoE enables us to compute a predicate over multiple inputs from several explicitly chosen clients. In SE, this would correspond to a search over documents where the query specifies which keywords have to be set by which parties. This is also the reason why existing multi-*writer* [7] schemes, do not consider searching over documents using queries which, for example, specify that party p_1 should have added keyword w_1, while party p_2 should have added keyword w_2.

2 Preliminaries

Throughout this paper, we use $x \overset{R}{\leftarrow} S$ to denote that x is chosen uniformly at random from the finite set S. We denote the ith component of a vector $\boldsymbol{v}$ as v_i. For a set of indices I, we write $\boldsymbol{v}_I$ for the subvector of $\boldsymbol{v}$. Instead of consistently using the vector notation, we use set notation when this is more convenient.

2.1 Primitives and Assumptions

Our construction uses *asymmetric bilinear maps*.

Definition 1 (Bilinear Map). *Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic multiplicative groups of prime order p. The map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an asymmetric bilinear map if the following two conditions hold.*

- *The map is bilinear; $\forall g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2,\ a, b \in \mathbb{Z}_p\colon e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.*
- *The map is non-degenerate; generators g_1 and g_2 are chosen such that the order of the element $e(g_1, g_2) \in \mathbb{G}_T$ equals p, the order of group $\mathbb{G}_T$.*

More specifically, we use a Type 3 pairing [13], where no efficiently computable homomorphisms between the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ can be found.

We use the function $\mathcal{G}(1^\kappa)$ to generate the parameters for a Type 3 bilinear group for the security parameter κ.

Additionally, we use a *pseudorandom permutation* (PRP) over $\mathcal{M} \subseteq \mathbb{Z}_p$.

Definition 2 (Pseudorandom Function). *For key space $\mathcal{K}$ and message space $\mathcal{M}$ define the function $\pi\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$. The function π is a pseudorandom permutation (PRP) if the output of π is indistinguishable from the output of a permutation chosen uniformly at random from the set of all possible permutations over $\mathcal{M}$.*

The security of our construction is based on the *decisional Diffie–Hellman* (DDH) problem and the *symmetric external Diffie–Hellman* (SXDH) problem.

Assumption 1. *The decisional Diffie–Hellman (DDH) assumption states that, given $(\mathbb{G}, g \in \mathbb{G}, g^a, g^b, Z)$ for uniformly at random chosen a and b, it is hard to distinguish $Z = g^{ab}$ from $Z \overset{R}{\leftarrow} \mathbb{G}$.*

Assumption 2. *Given the bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$, the symmetric external Diffie–Hellman (SXDH) assumption states that the DDH problem in both group $\mathbb{G}_1$ and group $\mathbb{G}_2$ is hard.*

3 Multi-client Predicate-Only Encryption

A multi-client predicate-only encryption scheme is a collection of the following four polynomial-time algorithms.

$\mathsf{Setup}(1^\kappa, n)$. This algorithm defines the public parameters pp, a master secret key msk, and the encryption keys usk_i for every client $1 \le i \le n$. The algorithm also defines the finite message space $\mathcal{M}^n$ and the predicate family $\mathcal{F}$, which predicates are efficiently computable on $\mathcal{M}^n$.

$\mathsf{Encrypt}(\mathsf{usk}_i, \mathsf{id}, x_i)$. A client i can encrypt a value $x_i \in \mathcal{M}$ using its encryption key usk_i and an identifier id. Different clients can use the same identifier, however, each client can only use an identifier at most once. The algorithm returns a ciphertext $\mathsf{ct}_{\mathsf{id},i}$. We usually omit the index id when there is no ambiguity. Furthermore, we introduce the following simplification of notation for a set of ciphertexts associated with the same id: For an ordered set $S \subseteq \{1, \ldots, n\}$ of indices, we write the set of ciphertexts $\{\, \mathsf{Encrypt}(\mathsf{usk}_j, \mathsf{id}, x_j) \mid j \in S \}$ as $\mathsf{Encrypt}(\mathsf{usk}_S, \mathsf{id}, \boldsymbol{x}_S)$. If $S = \{1, \ldots, n\}$, we simply write $\mathsf{Encrypt}(\mathsf{usk}, \mathsf{id}, \boldsymbol{x})$ or $\mathsf{ct}_{\boldsymbol{x}}$.

$\mathsf{GenToken}(\mathsf{msk}, f)$. The key generator can create a decryption key, termed *token*, for predicate $f \in \mathcal{F}$ using the msk. The algorithm returns the token tk_f.

$\mathsf{Test}(\mathsf{tk}_f, \mathsf{ct}_{\boldsymbol{x}})$. The Test algorithm requires a vector of ciphertexts $\mathsf{ct}_{\boldsymbol{x}}$ and a token tk_f as input. The algorithm outputs a Boolean value.

Definition 3 (Correctness). *A multi-client predicate-only encryption scheme is correct if $\mathsf{Test}(\mathsf{tk}_f, \mathsf{ct}_{\boldsymbol{x}}) = f(\boldsymbol{x})$. Formally, we require for all $n \in \mathbb{N}$, $\boldsymbol{x} \in \mathcal{M}^n$, and $f \in \mathcal{F}$,*

$$\Pr\left[\mathsf{Test}(\mathsf{ct}_{\boldsymbol{x}}, \mathsf{tk}_f) \neq f(\boldsymbol{x}) : \begin{array}{r} (\mathsf{pp}, \mathsf{msk}, \{\mathsf{usk}_i\}) \leftarrow \mathsf{Setup}(1^\kappa, n) \\ \mathsf{ct}_{\boldsymbol{x}} \leftarrow \mathsf{Encrypt}(\mathsf{usk}, \mathsf{id}, \boldsymbol{x}) \\ \mathsf{tk}_f \leftarrow \mathsf{GenToken}(\mathsf{msk}, f) \end{array}\right]$$

is negligible in the security parameter κ, where the probability is taken over the coins of Setup, $\mathsf{Encrypt}$, and $\mathsf{GenToken}$.

Note that we do not impose any restriction on the output of Test if it operates on messages encrypted under different identifiers.

3.1 Security

A commonly considered security game for private-key functional encryption is an indistinguishability-based notion under which the adversary may query both the $\mathsf{Encrypt}$ and the $\mathsf{GenToken}$ oracles [15,17,30]. Since our MC-POE is a special case of MC-FE, we start from the security notion from Goldwasser et al. [15]. However, they only consider the indistinguishability of plaintexts (*plaintext privacy* [17, 30]) and not of functions (*function* or *predicate privacy* [8,30]) in their security definition. In the following *full security* notion, we combine the plaintext-privacy and predicate-privacy notions, similarly to Shen et al. [30].

Because an evaluation of a predicate on a set of messages reveals some information about the messages in relation to the predicate (and vice versa), we cannot allow the adversary to query for all combinations of messages and predicates. For example, an adversary can distinguish an encryption of message $\boldsymbol{x}_0$ from an encryption of $\boldsymbol{x}_1$ if it has a token for a predicate f such that $f(\boldsymbol{x}_0) \neq f(\boldsymbol{x}_1)$. Even if we require $f(\boldsymbol{x}_0) = f(\boldsymbol{x}_1)$ for all predicates f that the adversary queried, a similar situation can still appear. To see this, consider an adversary corrupting client i so that it can encrypt any message m_i as ith input. This means that the adversary can also trivially distinguish the two messages if there exists a value m_i, such that if it replaces the ith input of $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ by m_i (resulting in inputs $\boldsymbol{x}'_0$ and $\boldsymbol{x}'_1$ respectively), the predicate has different outputs, i.e., $f(\boldsymbol{x}'_0) \neq f(\boldsymbol{x}'_1)$. Likewise, we also have to require that the predicates f_0 and f_1 yield the same result on a queried input $\boldsymbol{x}$, even if the adversary replaces some of the corrupted clients' inputs by another value.

In our security definition, we use the term *static corruptions* to indicate that the adversary announces the corrupted clients at the beginning of the game and cannot corrupt additional clients during the rest of the game. We let I be the set of indices of the uncorrupted clients and, similarly, indicate the indices of the corrupted clients by the set $\overline{I}$. Recall that we use the notation $\boldsymbol{x}_I$ to denote the subvector of $\boldsymbol{x}$ containing only the components from the set I. We denote with $f(\boldsymbol{x}_I, \cdot)$ a predicate f with the pre-filled inputs $\boldsymbol{x}_I$.

Definition 4 (Full Security). *A multi-client predicate-only encryption scheme is adaptive full secure under static corruptions if every probabilistic polynomial time adversary $\mathcal{A}$ has at most a negligible advantage in winning the following game.*

Initialization. *The adversary $\mathcal{A}$ submits a set of indices $\overline{I}$ to the challenger. We define the complement set $I = \{1, \ldots, n\} \setminus \overline{I}$.*

Setup. *The challenger runs $\mathsf{Setup}(1^\kappa, n)$ to get the pp, msk, and $\{\mathsf{usk}_i\}_{1 \leq i \leq n}$. It gives the public parameters pp and corrupted clients' keys $\{\, \mathsf{usk}_i \mid i \in \overline{I} \,\}$ to the adversary.*

Query 1. *The adversary $\mathcal{A}$ may query the challenger for ciphertexts or tokens.*

- *In case of a ciphertext query for (i, id, x_i), the challenger returns $\mathsf{ct}_{\mathsf{id},i} \leftarrow \mathsf{Encrypt}(\mathsf{usk}_i, \mathsf{id}, x_i)$.*
- *In case of a token query for f, the challenger returns $\mathsf{tk}_f \leftarrow \mathsf{GenToken}(\mathsf{msk}, f)$.*

Challenge. *The challenger picks a random bit b. The adversary can either request a ciphertext challenge or a token challenge.*

- *In case of a ciphertext challenge, the adversary sends $(\mathsf{id}^*, \boldsymbol{x}^*_{0,I}, \boldsymbol{x}^*_{1,I})$ to the challenger. The challenger returns the challenge $\mathsf{Ch}_I \leftarrow \mathsf{Encrypt}(\mathsf{usk}_I, \mathsf{id}^*, \boldsymbol{x}^*_{b,I})$.*

- *In case of a token challenge, the adversary sends* (f_0^*, f_1^*) *to the challenger. The challenger returns the challenge* $\mathsf{Ch} \leftarrow \mathsf{GenToken}(\mathsf{msk}, f_b^*)$.

Query 2. *The adversary may query the challenger again, similar to Query 1.*

Guess. *The adversary outputs its guess* $b' \in \{0, 1\}$ *for the bit* b.

We say that adversary $\mathcal{A}$ *wins the game, if* $b' = b$ *and*

- *in case of a ciphertext challenge,* $\mathcal{A}$ *did not query for a ciphertext using identifier* id^* *in any of the two query phases, nor query for a predicate* f, *such that* $f(\boldsymbol{x}_{0,I}^*, \cdot) \neq f(\boldsymbol{x}_{1,I}^*, \cdot)$;
- *in case of a token challenge,* $\mathcal{A}$ *did not query for* (i, id, x_i), *for uncorrupted clients* $i \in I$, *such that it can combine these inputs* x_i *for the same* id, *into a vector* $\boldsymbol{x}_I$, *where* $f_0^*(\boldsymbol{x}_I, \cdot) \neq f_1^*(\boldsymbol{x}_I, \cdot)$.

Note that in the above defined game, in case of a ciphertext challenge, the challenger only returns challenge ciphertexts for the uncorrupted clients. The adversary can still evaluate predicates on the received challenge by generating the ciphertext values for the corrupted clients using their encryption keys.

It is important to realize that the challenger can decide whether the adversary wins the game or not in polynomial time. This is possible because the adversary $\mathcal{A}$ can only query for a polynomial number of ciphertexts and tokens. Moreover, the challenger is able to efficiently check if $f(\boldsymbol{x}_I, \cdot) = f'(\boldsymbol{x}'_I, \cdot)$ as both n and $\mathcal{M}^n$ are finite and fixed by $\mathsf{Setup}(1^\kappa, n)$.

Definition 5 (Selective Full Security). *The definition of a selective full secure under static corruptions multi-client predicate-only encryption scheme is similar to the adaptive full security notion of Definition 4. The difference between the two, is that in selective security game, the challenge request (i.e.,either* $(\mathsf{id}^*, \boldsymbol{x}_{0,I}^*, \boldsymbol{x}_{1,I}^*)$ *or* (f_0^*, f_1^*)*) is announced during Initialization.*

As explained before, the full security definition actually defines two security notions. We say that MC-PoE scheme is *adaptive (selective) plaintext private* if no adversary can win the adaptive (selective, respectively) full security game with a ciphertext challenge. Similarly, PoE scheme is *adaptive (selective) predicate private* if no adversary can win the adaptive (selective, respectively) full security game with a token challenge.

Chosen-Plaintext Security. The definition of full security is very strong as it allows an adversary to query for both ciphertexts and tokens. This is similar to the chosen-ciphertext attack (CCA) security notion used in public-key cryptography, where the adversary can query both the encryption and decryption[2] oracle. To accommodate for a different attacker model, we define a *chosen-plaintext* security notion, where the adversary only has access to the encryption oracle and is asked to distinguish between two ciphertexts. Such a notion is similar to

---

[2] In MC-PoE, an adversary can use a token and the public Test algorithm to learn more about the encrypted plaintext.

chosenplaintext attack (CPA) security as defined in public-key cryptography and is also related to the *offline security* notion of Lewi and Wu [18], in which an attacker has only access to ciphertexts and not to decryption keys. To make our notion stronger, we give the adversary access to all clients' encryption keys (but not to the internal randomness of the clients).

Definition 6 (Chosen-Plaintext Security). *A multi-client predicate-only encryption scheme is chosen-plaintext secure under unbounded corruptions if any probabilistic polynomial time algorithm $\mathcal{A}$ has at most a negligible advantage in winning the following game.*

Setup. *The challenger runs* $\mathsf{Setup}(1^\kappa, n)$ *to get the* pp, msk, *and* $\{\mathsf{usk}_i\}_{1 \leq i \leq n}$. *It gives the public parameters* pp *and all clients' keys* $\{\mathsf{usk}_i\}_{1 \leq i \leq n}$ *to the adversary. Note that the adversary* $\mathcal{A}$ *can encrypt any message* x_i *for identifier* id *using the key* usk_i *by computing* $\mathsf{Encrypt}(\mathsf{usk}_i, \mathsf{id}, x_i)$.

Challenge. *The adversary sends the challenge request* $(\mathsf{id}^*, \boldsymbol{x}_0^*, \boldsymbol{x}_1^*)$ *to the challenger. The challenger picks a random bit* b *and returns* $\mathsf{Encrypt}(\mathsf{usk}, \mathsf{id}^*, \boldsymbol{x}_b^*)$ *to the adversary.*

Guess. *The adversary outputs its guess* $b' \in \{0, 1\}$ *for the bit* b.

We say that adversary $\mathcal{A}$ *wins the game if* $b' = b$.

Observe that in this game the adversary is given *every* client's private key. This security requirement is quite strong and corresponds to a following situation: Even if an attacker compromises a client and steals its encryption keys, it *remains* hard for the attacker to determine the plaintexts of the ciphertexts created *before and after* the compromise.

4 Our Construction

We construct a multi-client predicate-only encryption scheme for the functionality of a conjunctive equality test. To test if n messages $x_1, \dots, x_n$, encrypted by distinct clients, equal the values $y_1, \dots, y_n$, we evaluate the predicate

$$\mathsf{Match}(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \text{TRUE} & \text{if } \bigwedge_{i=1}^n (x_i = y_i), \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

As discussed in Sect. 1.1, this functionality turns out to be surprisingly useful in the domain of critical infrastructure protection. In this setting, a monitor combines the ciphertexts associated with the same identifier and evaluates all its tokens (corresponding to various predicates) on the ciphertext vector to see if there is a match. If a match is found, the monitor may raise an alarm or take other appropriate actions. A schematic overview of relations among all parties of such a multi-client monitoring system is shown in Fig. 1.

We now describe our multi-client predicate-only encryption construction for conjunctive equality tests over multiple clients.

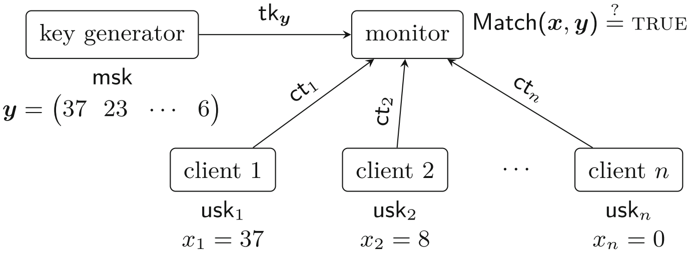

Fig. 1. In this example of a multi-client monitoring system, there are n distinct clients (with keys $\mathsf{usk}_1, \ldots, \mathsf{usk}_n$) that determine the values $x_1, \ldots, x_n$. The monitor computes the functionality $\mathsf{Match}(\boldsymbol{x}, \boldsymbol{y})$ using the encrypted values $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ and a token $\mathsf{tk}_{\boldsymbol{y}}$. The monitor is only able to compute the functionality if all clients encrypted their value x_i using the same identifier id (not shown in the figure).

Setup($1^\kappa, n$). Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\kappa)$ be the parameters for a bilinear group. Choose a pseudorandom permutation $\pi\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ for message space $\mathcal{M} \subseteq \mathbb{Z}_p$ and a cryptographic hash function $H\colon \{0,1\}^* \to \mathbb{G}_1$. The bilinear group parameters together with both functions form the public parameters. To generate the keys, select $\alpha_i, \gamma_i \xleftarrow{R} \mathbb{Z}_p^*$ and $\beta_i \xleftarrow{R} \mathcal{K}$ for $1 \le i \le n$. The master secret key is

$$\mathsf{msk} = \left\{ (g_2^{\alpha_i}, \beta_i, g_2^{\gamma_i}) \right\}_{i=1}^{n}.$$

The secret encryption key for client i is

$$\mathsf{usk}_i = (g_1^{\alpha_i}, \beta_i, \gamma_i).$$

Encrypt($\mathsf{usk}_i, \mathsf{id}, x_i$). Client i can encrypt its message $x_i \in \mathcal{M}$ for identifier id using usk_i and $r_i \xleftarrow{R} \mathbb{Z}_p^*$,

$$\mathsf{ct}_i = \left(H(\mathsf{id}), g_1^{r_i}, g_1^{\alpha_i \pi(\beta_i, x_i) r_i} H(\mathsf{id})^{\gamma_i} \right).$$

GenToken($\mathsf{msk}, \boldsymbol{y}$). The token generator can encrypt a vector $\boldsymbol{y} \in \mathcal{M}^n$ using its key msk. Choose $u_i \xleftarrow{R} \mathbb{Z}_p^*$ for $1 \le i \le n$ and output

$$\mathsf{tk}_{\boldsymbol{y}} = \left(\left\{ g_2^{u_i}, g_2^{\alpha_i \pi(\beta_i, y_i) u_i} \mid 1 \le i \le n \right\}, \prod_{1 \le i \le n} (g_2^{\gamma_i})^{u_i} \right).$$

Test($\mathsf{tk}_{\boldsymbol{y}}, \{\mathsf{ct}_i\}_{1 \le i \le n}$). Output the result of the test

$$\prod_{1 \le i \le n} e\left(g_1^{\alpha_i \pi(\beta_i, x_i) r_i} H(\mathsf{id})^{\gamma_i}, g_2^{u_i}\right) \overset{?}{=} \prod_{1 \le i \le n} e\left(g_1^{r_i}, g_2^{\alpha_i \pi(\beta_i, y_i) u_i}\right) e\left(H(\mathsf{id}), \prod_{1 \le i \le n} (g_2^{\gamma_i})^{u_i}\right).$$

4.1 Correctness

Correctness follows from the definition of Test. We remark that the output of Test is completely determined by $\sum_{1\leq i\leq n}\big(\pi(\beta_i, x_i) - \pi(\beta_i, y_i)\big) \overset{?}{=} 0$. Since the function π is a PRP, the probability of $\mathsf{Test}(\mathsf{tk}_{\boldsymbol{y}}, \mathsf{ct}_{\boldsymbol{x}}) \neq \mathsf{Match}(\boldsymbol{x}, \boldsymbol{y})$ is negligible.

4.2 Security

To get an intuition for the security of our construction, observe that the clients' messages itself are first encrypted using the PRP π. By using the output of the PRP as an exponent and randomizing it with the value r, we create a probabilistic encryption of the message. The PRP's randomized output also prevents malleability attacks. Similarly, the vector components of the vector $\boldsymbol{y}$ are individually encrypted in a similar way. Because part of the clients' keys (i.e., $g_1^{\alpha_i}$) and the master secret key (i.e., $g_2^{\alpha_i}$) reside in different groups, it is hard for a client to create a token and hard for the token generator to create a ciphertext.

The formal security analysis can be found in Appendix A. We prove our construction selective plaintext private and adaptive predicate private. Additionally, we prove the chosen-plaintext security property of the construction. Plaintext and predicate privacy are proven in the generic group model using random oracles. This combination of models has been successfully applied in other works before [9,34]. Chosen-plaintext security can be proven in the standard model and under the DDH assumption in group $\mathbb{G}_1$. We formulate the following two theorems.

Theorem 1. *Let $\mathcal{A}$ be an arbitrary probabilistic polynomial time adversary having oracle access to the group operations and the encryption and token generation algorithms, while it is bounded in receiving at most q distinct group elements. The adversary $\mathcal{A}$ has at most an advantage of $O(q^2/p)$ in winning either the selective plaintext-privacy (see Definition 5) or the adaptive predicate-privacy game (see Definition 4) in the random oracle model.*

Theorem 2. *The construction presented above is chosen-plaintext secure with an unbounded number of corruptions (Definition 6) under the DDH assumption in group $\mathbb{G}_1$.*

Both plaintext privacy and predicate privacy are proven secure through a series of hybrid games. In every game hop, a component of the challenge vector (either the ciphertext or token challenge vector) is replaced by a random one. In the final game, once all components are replaced by random elements, no adversary can gain an advantage since it is impossible to distinguish a random vector from another random one.

However, in the selective plaintext-privacy game, not *every* component of the challenge vector can be replaced by a random component. If a component $x_{b,i}^*$ of the challenge vector $\boldsymbol{x}_b^*$ is deterministic, i.e.,the challenge inputs were the same for that component, $x_{0,i}^* = x_{1,i}^* = m$, the adversary may query for a token to match this single component for the value $y_i = m$. Note that if this component is replaced by a random element, Match will, with overwhelming probability, return FALSE, while it should have returned TRUE. Hence, the deterministic components of the challenge vector have to remain untouched in every game hop. This implies that the number of game hops depends on the challenge inputs, requiring the challenger to know the challenge inputs a priori. This limitation does not appear for predicate privacy, making it possible to prove adaptive security instead.

4.3 Extension Allowing Wildcards

Although a construction for the described conjunctive equality matching functionality would suffice, it may be very inefficient when a predicate is defined over a subset of the clients' inputs. For example, suppose the token generator has a predicate for which it actually does not care what client i sends. Now, if we have only conjunctive equality matching, we would need to create a token for every possible message that client i can send. Besides that this will be very inefficient if client i could send many different messages, it would also reveal whenever client i has sent the same values multiple times: whenever a client sends the same value multiple times, the same token will match multiple times as well!

We can extend our construction with the ability to test for the equality of vectors with the additional feature that the predicate vector $\boldsymbol{y}$ can now contain wildcard components. Such a wildcard component matches against any value of the corresponding ciphertext component. This makes the testing functionality similar to the one used in HVE [6], however our system combines the ciphertexts from multiple clients. Formally, the clients encrypt their messages from the message space $\mathcal{M} \subseteq \mathbb{Z}_p$, where the token generator uses the space $\mathcal{M}^* = \mathcal{M} \cup \{\star\}$. The multi-client predicate-only encryption construction now evaluates the function

$$\mathsf{Match}^\star(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \text{TRUE} & \text{if } \forall i \colon (x_i = y_i) \vee (y_i = \star), \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

To achieve this additional functionality, we have to change the GenToken and Test algorithms, the other algorithms remain unchanged.

GenToken*(msk, $\boldsymbol{y}$). The token generator can encrypt a predicate vector $\boldsymbol{y} \in (\mathcal{M}^*)^n$ using the master secret key msk. Let $S_{\boldsymbol{y}}$ be the set of indices of the non-wildcard components of the vector $\boldsymbol{y}$. Choose $u_i \xleftarrow{R} \mathbb{Z}_p^*$ for $i \in S_{\boldsymbol{y}}$ and output

$$\mathsf{tk}_{\boldsymbol{y}} = \left(\left\{ g_2^{u_i}, g_2^{\alpha_i \pi(\beta_i, y_i) u_i} \mid i \in S_{\boldsymbol{y}} \right\}, \prod_{i \in S_y} (g_2^{\gamma_i})^{u_i} \right).$$

Test*(tk$_{\boldsymbol{y}}$, $\{\mathsf{ct}_i\}_i \in S_{\boldsymbol{y}}$). Output the result of the test

$$\prod_{i \in S_y} e\big(g_1^{\alpha_i \pi(\beta_i, x_i) r_i} H(\mathsf{id})^{\gamma_i}, g_2^{u_i}\big) \stackrel{?}{=} \prod_{i \in S_y} e\big(g_1^{r_i}, g_2^{\alpha_i \pi(\beta_i, y_i) u_i}\big) e\big(H(\mathsf{id}), \prod_{i \in S_y} (g_2^{\gamma_i})^{u_i}\big).$$

In this adapted construction, the wildcards are made possible by allowing the token generator to specify which clients need to contribute a ciphertext before one can evaluate the predicate over the subset of clients. This idea is encoded in the token by the value $\prod_{i \in S_y} (g_2^{\gamma_i})^{u_i}$ and in the ciphertext by the value $H(\mathsf{id})^{\gamma_i}$. The latter also prevents the monitor to combine ciphertext for different identifiers.

The addition of wildcards to the scheme should be mainly considered an efficiency improvement, rather than a security improvement, although the ciphertext security actually slightly improves when one uses wildcards – the wildcard components do not leak any information about the matched ciphertext, as discussed above. However, we point out that this adapted construction is not predicate private. In fact, if wildcards are used in the proposed construction, the token would leak their positions: by looking at a token, it is possible to tell which components encode a wildcard. But, if we accept this fact, yet still want to assure that no other information is leaked, we can define a *restricted* predicate-privacy game. In this restricted game, we restrict the adversary to only provide challenge inputs with wildcards in the same position, i.e.,we require for challenge inputs $f_0^* = \boldsymbol{y}_0^*$, $f_1^* = \boldsymbol{y}_1^*$ that for all $1 \leq i \leq n$, $y_{0,i} = \star \iff y_{1,i} = \star$.

It is trivial to see that changing the GenToken or Test algorithm does not influence the chosen-ciphertext security. In Appendix A we give the security proofs for the construction with wildcards.

4.4 Efficiency

Since the Encrypt and GenToken algorithms do not use any expensive pairing operations, they can efficiently run on less powerful hardware. For the Encrypt algorithm it is only needed to compute the PRP π and three modular exponentiations. The computational complexity of GenToken* depends on the number of non-wildcard components in the predicate. For every non-wildcard component one evaluation of the PRP π and three modular exponentiations are needed.

The Test algorithm is the only algorithm that requires pairings. To evaluate a token with n non-wildcard components, $2n + 1$ pairing evaluations are needed.

In the next section we discuss a concrete implementation of the construction and evaluate its performance.

5 Implementation and Evaluation

We have implemented a prototype of our construction with wildcards to get a better understanding of its performance. The implementation[3] uses the Pairing-Based Cryptography Library[4] that allows one to easily change the underlying curve and its parameters.

Instantiating the Pseudorandom Permutation. Our construction uses a PRP π to permute an element in $\mathbb{Z}_p$. However, since we use the outcome of the permutation to exponentiate a generator in $\mathbb{G}_1$ and $\mathbb{G}_2$, we can instead directly map values in $\mathbb{Z}_p$ to one of these groups respectively. The pseudorandom function (PRF) proposed by Naor and Reingold [24] exactly achieves this. Their PRF maps a message $x \in \mathcal{M} \subseteq \{0, \ldots, 2^m - 1\} \subseteq \mathbb{Z}_p$ using a key $\boldsymbol{b} = \left\{ b_i \xleftarrow{R} \mathbb{Z}_p^* \mid 0 \leq i \leq m \right\}$ to an element in a group $\langle g \rangle$ of prime order p. The PRF F is defined as

$$F(\boldsymbol{b}, x) = g^{b_0 \prod_{i=1}^{m} b_i^{x[i]}},$$

where $x[i] \in \{0, 1\}$ denotes the ith bit of message x. The advantage of using this PRF over PRP is that it is relatively simple to compute while it is provably secure under the DDH assumption.

We apply the PRF to both the $\mathsf{Encrypt}$ and the $\mathsf{GenToken}^\star$ algorithms to obtain ciphertexts of the form

$$\mathsf{ct}_i = \left(H(\mathsf{id}), g_1^{r_i}, g_1^{\alpha_i \prod_{j=1}^{m} \beta_{i,j}^{x_i[j]} r_i} H(\mathsf{id})^{\gamma_i} \right),$$

and tokens of the form

$$\mathsf{tk}_{\boldsymbol{y}} = \left(\left\{ g_2^{u_i}, g_2^{\alpha_i \prod_{j=1}^{m} \beta_{i,j}^{y_i[j]} u_i} \mid i \in S_{\boldsymbol{y}} \right\}, \prod_{i \in S_{\boldsymbol{y}}} (g_2^{\gamma_i})^{u_i} \right).$$

Notice that we use $b_0 = \alpha_i$ and $b_j = \beta_{i,j}$. In addition, observe that it is not necessary to know the value α_i to compute a ciphertext or token, as long the value $g_1^{\alpha_i}$, or $g_2^{\alpha_i}$ respectively, is known.

Performance Measurements. We ran several performance evaluations on a notebook containing an Intel Core i5 CPU, running on Debian GNU/Linux. We chose to evaluate the system using an MNT curve [22] over a 159 bit base field size with embedding degree 6.

As expected from the theoretical performance analysis in Sect. 4.4, both the $\mathsf{GenToken}^\star$ and $\mathsf{Test}^\star$ algorithms scale linearly in the number of non-wildcard components used. The $\mathsf{GenToken}^\star$ algorithm spends, on average, 19 ms to encrypt a non-wildcard component. To evaluate a token that contains no wildcards using

---

[3] https://github.com/CRIPTIM/multi-client-monitoring.

[4] https://crypto.stanford.edu/pbc/.

n ciphertexts, takes $4.5n + 10$ ms on average. The Setup algorithm scales linearly as well, spending on average 18 ms per client to create their public and private keys. The Encrypt algorithm is the fastest, taking only 2.6 ms for an individual client to encrypt a message $x_i \in \{0, \ldots, 15\}$.

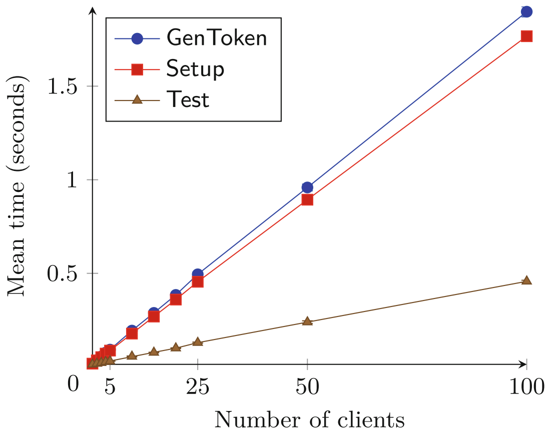

Fig. 2. Performance measurements of the implementation using an MNT-159 curve.

In Fig. 2 the average computational time is plotted against the number of clients involved in the computation. No wildcards were used in the GenToken* and Test* algorithms to obtain these timing results, meaning that the algorithms are identical to GenToken and Test, respectively.

Considering an example of the monitoring of several CIs, we remark that a typical information-sharing community (e.g., ISAC) consists of about 10 parties. So, if every party sends 5 distinct messages for each identifier (e.g., every party has five subsystems to be monitored), we would require a system of about 50 clients. We see that in such a realistically sized system we can evaluate about 250 predicates per minute. Optimizations such as the preprocessing of pairings can increase the number of predicate evaluations per minute.

6 Conclusion

By designing a special-purpose multi-client functional encryption scheme, it is possible to create a practical privacy-preserving monitoring system. To achieve this, we defined multi-client predicate-only encryption (MC-PoE) and corresponding security definitions for the protection of both the messages of the individual clients and the predicates. Our proposed construction for such PoE scheme is capable of conjunctive equality testing over vector components which can include wildcards. The performance evaluation of our implementation shows that the evaluation time of a predicate scales linearly in the number of clients, where a predicate defined over 20 clients can be evaluated in a tenth of a second. Additionally, we see that the encryption algorithm is very lightweight, making it suitable to run on resource-constrained devices.

Future work will include the construction of MC-PoE scheme which will allow for more expressive functionality, while remaining efficient enough to run in practice and keeping the confidentiality of both the messages and the predicates. Additionally, further research is needed to construct MC-PoE scheme that is fully secure in the standard model.

Acknowledgment. This work was supported by the Netherlands Organisation for Scientific Research (NWO) in the context of the CRIPTIM project. The authors additionally thank the reviewers and shepherd for their suggested improvements.

A Security Proofs

A.1 Selective Plaintext and Adaptive Predicate Security

We prove Theorem 1, stating that the construction without wildcards is secure, by using the following lemma and by proving that the construction with wildcards is selective plaintext private and restricted adaptive predicate private. Recall that the restricted predicate-private game is almost identical to our predicate-private game. However, in the restricted game, we additionally require $y^*_{0,i} = \star \iff y^*_{1,i} = \star$ for the challenge inputs $\boldsymbol{y}^*_0, \boldsymbol{y}^*_1$.

Lemma 1. *If the construction* with *wildcards is selective plaintext private and* restricted *adaptive predicate private, then the construction* without *wildcards is selective plaintext private and adaptive predicate private.*

Proof. First, let us look at the selective plaintext privacy. Assume $\mathcal{A}$ is a probabilistic polynomial time adversary, having a non-negligible advantage in winning the selective plaintext-privacy game without wildcards. It is clear that $\mathcal{A}$ is also an adversary that has an identical, non-negligible, advantage in winning the selective plaintext-privacy game with wildcards (however, it chooses not to use any). This contradicts with the given statement that no such adversary exists.

For the other part, assume that $\mathcal{A}$ is a probabilistic polynomial time adversary, making no wildcard queries, and having a non-negligible advantage in winning the predicate-privacy game. Note that $\mathcal{A}$ is also an adversary that has an identical, non-negligible, advantage in winning the predicate-privacy game with wildcards (however, it chooses not to use any). Specifically, since $\mathcal{A}$ chooses its challenge inputs without wildcards, $\mathcal{A}$ also satisfied the extra requirement in the restricted predicate-privacy game.

We now give a proof for both selective plaintext privacy as well as restricted predicate privacy for the construction with wildcards.

Proof (sketch). We first define the generic group model setting and all oracle interactions, including the oracles for encryption and token generation.

Generic Group Model. Let ϕ_1, ϕ_2, ϕ_T be distinct random injective mappings from the domain $\mathbb{Z}_p$ to $\{0,1\}^\kappa$, where $\kappa > 3\log p$. We write $\mathbb{G}_1$ for $\{\, \phi_1(x) \mid x \in \mathbb{Z}_p \,\}$, $\mathbb{G}_2$ for $\{\, \phi_2(x) \mid x \in \mathbb{Z}_p \,\}$, and $\mathbb{G}_T$ for $\{\, \phi_T(x) \mid x \in \mathbb{Z}_p \,\}$. The adversary is given access to an oracle to compute the group actions on $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Additionally, it is given access to an oracle capable of computing a non-degenerate bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Lastly, we also define a random oracle to model the hash function $H\colon \{0,1\} \to \mathbb{G}_1$.

Instead of writing $\phi_1(x)$, we write g_1^x. Similarly, we write g_2^x for $\phi_2(x)$ and $e(g_1, g_2)^x$ for $\phi_T(x)$.

Hash Oracle H. The challenger keeps track of oracle queries it received before by maintaining a table. If it has not received an oracle query for the value id before, it chooses a random value $t_{\mathsf{id}} \in \mathbb{Z}_p$ and stores this value in its table. It returns the value $g_1^{t_{\mathsf{id}}}$ to the querier.

Game Interactions. The adversary's first interaction with the challenger is to receive the group parameters and the secret keys of the corrupted clients.

Setup. The challenger chooses $\alpha_i, \gamma_i \xleftarrow{R} \mathbb{Z}_p^*$ and $\beta_i \xleftarrow{R} \mathcal{K}$ for $1 \le i \le n$, just like in the actual scheme. It also defines the secret keys usk_i and master secret key msk according to the scheme.

Corruptions. The adversary submits its choices for the corrupted clients $\overline{I}$ to the challenger. In the selective plaintext-privacy game, the adversary additionally submits its challenge inputs $(\mathsf{id}^*, \boldsymbol{x}^*_{0,I}, \boldsymbol{x}^*_{1,I})$. The challenger gives the secret keys $\mathsf{usk}_{\overline{I}}$ of the corrupted clients to the adversary.

Queries. The adversary interacts with the challenger by asking the challenger to encrypt a messages or to generate a token for some predicate. To be able to refer to a specific query later on in the proof, we label every query with a query number. Let j represent this query number.

Encrypt The challenger answers valid $\mathsf{Encrypt}$ queries for a message $x_i^{(j)}$ for client i and identifier $\mathsf{id}^{(j)}$ similar as in the scheme. It chooses $r_i^{(j)} \xleftarrow{R} \mathbb{Z}_p^*$ and returns the ciphertext $\mathsf{ct}^{(j)}_{i,\mathsf{id}}$,

$$\left(g_1^{t_{\mathsf{id}^{(j)}}}, g_1^{r_i^{(j)}}, g_1^{\alpha_i \pi(\beta_i, x_i^{(j)}) r_i^{(j)}} g_1^{t_{\mathsf{id}^{(j)}} \gamma_i} \right).$$

GenToken* Similarly, token queries for $\boldsymbol{y}^{(j)}$ are answered according to the scheme as well. The challenger chooses $u_i^{(j)} \xleftarrow{R} \mathbb{Z}_p^*$ for $i \in S_{\boldsymbol{y}^{(j)}}$ and returns the token $\mathsf{tk}^{(j)}_{\boldsymbol{y}}$,

$$\left(\left\{ g_2^{u_i^{(j)}}, g_2^{\alpha_i \pi(\beta_i, y_i^{(j)}) u_i^{(j)}} \mid i \in S_{\boldsymbol{y}} \right\}, \prod_{i \in S_{\boldsymbol{y}}} g_2^{u_i^{(j)} \gamma_i} \right),$$

to the adversary.

Proof Structure. We prove both selective plaintext privacy and restricted adaptive predicate privacy through a series of hybrid games.

For selective plaintext privacy the number of games depends on the number of differentiating components of the challenge inputs – hence the *selective* game type. Let $\overline{X}$ denote the set of indices where the components of $\boldsymbol{x}_0^*$ differ from $\boldsymbol{x}_1^*$, $\overline{X} = \{\, i \mid x_{0,i}^* \neq x_{1,i}^* \,\}$. Let game k be identical to the original game, except that in the challenge phase now the first $k-1$ components of $\overline{X}$ in the returned challenge vector are chosen at random. Note that game $k=1$ is identical to the original game and that in game $k = |\overline{X}|$ not even an unbounded adversary is able to gain an advantage in winning the game.

For restricted adaptive predicate privacy, we assume w.l.o.g. that $\boldsymbol{y}_{0,I}^* \neq \boldsymbol{y}_{1,I}^*$, because if $\boldsymbol{y}_{0,I}^* = \boldsymbol{y}_{1,I}^*$, the adversary would not be able to gain an advantage in the game since this implies $\boldsymbol{y}_0^* = \boldsymbol{y}_1^*$. Note that this means that the result of $\mathsf{Match}^\star$ with any allowed ciphertext vector will be FALSE. We define game k identical to the original game, except that in the challenge phase now the first $k-1$ components of the returned challenge vector are chosen at random. Note that game $k=1$ is identical to the original game and that in game $k=n$ not even an unbounded adversary is able to gain an advantage in winning the game.

For both the selective plaintext-privacy as well as the restricted adaptive predicate-privacy game, we show that an adversary has at most an advantage of $O(q^2/p)$ in distinguishing between game k and game $k+1$. Furthermore, we use another hybrid game to change to a real-or-random based challenge instead of a left-or-right based challenge. It is not difficult to see that an adversary gaining an advantage ϵ in the left-or-right based game, gains an advantage of at least $\frac{\epsilon}{2}$ in the real-or-random based game.

Challenges. Since we changed the game to a real-or-random based game, the challenge phase changes slightly. The challenger now chooses a bit $b \xleftarrow{R} \{0,1\}$ that is used to determine whether to return the encryption of the submitted value or a random one. In case of the selective plaintext-privacy game, the adversary submits a vector $\boldsymbol{x}_I^{(c)}$ together with an identifier $\mathsf{id}^{(c)}$ to the challenger. In case of the restricted predicate-privacy game, the adversary submits a vector $\boldsymbol{y}^{(c)}$ to the challenger. The challenger chooses values $\nu_i, \nu_i' \xleftarrow{R} \mathbb{Z}_p^*$ for $1 \leq i \leq n$. For a ciphertext challenge it returns the challenge

$$\mathsf{ct}_{\mathsf{Ch}} = \left\{ \left(g_1^{t_{\mathsf{id}^{(c)}}}, g_1^{\nu_i}, \mathsf{ct}'_{\mathsf{Ch},i}\right) \mid i \in I \right\},$$

where

$$\mathsf{ct}'_{\mathsf{Ch},k} = \begin{cases} g_1^{\nu_k \alpha_k \pi(\beta_k, \nu_k') + t_{\mathsf{id}^{(c)}} \gamma_k} & \text{if} \quad b = 0 \\ g_1^{\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{id}^{(c)}} \gamma_k} & \text{if} \quad b = 1. \end{cases}$$

For a token challenge, it returns the challenge

$$\mathsf{tk}_{\mathsf{Ch}} = \left(\left\{ \, (g_2^{\nu_i}, \mathsf{tk}'_{\mathsf{Ch},i}) \mid i \in S_{\boldsymbol{y}} \, \right\}, \prod_{i \in S_y} g_2^{\nu_i \gamma_i} \right),$$

where, if $k \in S_{\boldsymbol{y}}$,

$$\mathsf{tk}'_{\mathsf{Ch},k} = \begin{cases} g_2^{\nu_k \alpha_k \pi(\beta_k, \nu'_k)} & \text{if } b = 0 \\ g_2^{\nu_k \alpha_k \pi(\beta_k, y_k^{(c)})} & \text{if } \ b = 1. \end{cases}$$

Indistinguishability. We now show that an adversary has at most a negligible advantage of $O(q^2/p)$ in distinguishing between game k and game $k+1$, i.e.,it is unable to distinguish $g_1^{\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{id}(c)} \gamma_k}$ from $g_1^{\nu'_k}$ for ciphertext challenges and $g_2^{\nu_k \alpha_k \pi(\beta_k, y_k^{(c)})}$ from $g_2^{\nu'_k}$ for token challenges.

As is common in the generic bilinear group model [32], we consider the challenger keeping record of all group elements the adversary has. It does so by keeping lists $P_{\mathbb{G},l}$ of linear polynomials in $\mathbb{Z}_p$ for each of the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. These polynomials use indeterminates for γ_i, $\alpha_i \pi(\beta_i, c_i)$, the $t_{\mathsf{id}^{(j)}}$'s, $\alpha_i \pi(\beta_i, x_i^{(j)})$'s, $\alpha_i \pi(\beta_i, y^{(j)})$'s, $r_i^{(j)}$'s, and the $u_i^{(j)}$'s.

To simplify our reasoning, we will only look at polynomials $P_{\mathbb{G}_T,l}$ in $\mathbb{G}_T$. This is justified as we can transform any polynomial in $\mathbb{G}_1$ or $\mathbb{G}_2$ to a polynomial $P_{\mathbb{G}_T,l}$ in $\mathbb{G}_T$ through an additional query to the pairing oracle.

We can now say that the adversary wins the game if for a random assignment to all the indeterminates, any $P_{\mathbb{G}_T,i} \neq P_{\mathbb{G}_T,j}$ evaluates to the same value. We will show that the adversary is not able to query for distinct polynomials $P_{\mathbb{G}_T,i}$, $P_{\mathbb{G}_T,j}$ such that, if the challenger plays the 'real' experiment and if the indeterminates get assigned with random values, they will evaluate to the same value, except for negligible probability. Then, by the Schwartz lemma [29] and the extended result of Shoup [32], we can bound this probability of $P_{\mathbb{G}_T,i} \neq P_{\mathbb{G}_T,j}$ evaluating to the same value by $O(q^2/p)$ if at most q group elements are given to the adversary.

In the case of a ciphertext challenge, we first have to bring the challenge response, which is an element of $\mathbb{G}_1$, to the target group $\mathbb{G}_T$. Since the adversary only has (linear combinations of) the elements g_2, $g_2^{u_i^{(j)}}$, $g_2^{\alpha_i \pi(\beta_i, y_i^{(j)}) u_i^{(j)}}$, and $\prod_{i \in S_y} g_2^{u_i^{(j)} \gamma_i}$ in $\mathbb{G}_2$, it can only bring the challenge to $\mathbb{G}_T$ by pairing with one of these. Similarly, for token challenges, the adversary can only pair with the elements g_1, $g_1^{t_{\mathsf{id}(j)}}$, $g_1^{r_i^{(j)}}$, or $g_1^{\alpha_i \pi(\beta_i, x_i^{(j)}) r_i^{(j)} + t_{\mathsf{id}(j)} \gamma_i}$ in $\mathbb{G}_1$.

The resulting polynomials for these challenge responses are summarized in Table 1. Since the group elements are represented by uniformly independent values, the adversary can only distinguish between game k and game $k+1$ with more than a negligible advantage if it can construct at least one of the polynomials in this table.

Table 1. Target polynomials in both indistinguishability games.

Ciphertext challenge

$\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{id}^{(c)}} \gamma_k$	$u_i^{(j)} (\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{id}^{(c)}} \gamma_k)$
$u_i^{(j)} \alpha_i \pi(\beta_i, y_i^{(j)}) (\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{id}^{(c)}} \gamma_k)$	$(\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{id}^{(c)}} \gamma_k) \sum_{i \in S_{\boldsymbol{y}^{(j)}}} u_i^{(j)} \gamma_i$

Token challenge

$\nu_k \alpha_k \pi(\beta_k, y_k^{(c)})$	$t_{\mathsf{id}} \nu_k \alpha_k \pi(\beta_k, y_k^{(c)})$
$r_i^{(j)} \nu_k \alpha_k \pi(\beta_k, y_k^{(c)})$	$\left(r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{id}} \gamma_i\right) \nu_k \alpha_k \pi(\beta_k, y_k^{(c)})$

Linear Combinations. We now argue that the adversary cannot construct any of these challenges by looking at the components it has. We summarize the polynomials the adversary has access to, again by only looking at the elements in the target group $\mathbb{G}_T$, in Table 2.

Table 2. Elements the adversary can query for in an indistinguishability game (up to linear combinations).

1	$t_{\mathsf{id}^{(j)}}$
$u_{i'}^{(j')}$	$u_{i'}^{(j')} t_{\mathsf{id}^{(j)}}$
$u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')})$	$u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')}) t_{\mathsf{id}^{(j)}}$
$\sum_{i' \in S_{\boldsymbol{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$	$t_{\mathsf{id}^{(j)}} \sum_{i' \in S_{\boldsymbol{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$

$\dots$

$\dots$

$r_i^{(j)}$	$r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{id}^{(j)}} \gamma_i$
$u_{i'}^{(j')} r_i^{(j)}$	$u_{i'}^{(j')} \left(r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{id}^{(j)}} \gamma_i\right)$
$u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')}) r_i^{(j)}$	$u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')}) \left(r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{id}^{(j)}} \gamma_i\right)$
$r_i^{(j)} \sum_{i' \in S_{\boldsymbol{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$	$\left(r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{id}^{(j)}} \gamma_i\right) \sum_{i' \in S_{\boldsymbol{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$

We show in the full version of the paper that no linear combination of the polynomials in Table 2 equals any of the polynomials in Table 1.

A.2 Chosen-Plaintext Security

The proposed construction is also chosen-plaintext secure as stated in Theorem 2. We remark that the proof does not rely on the use of random oracles.

Proof. We construct a challenger $\mathcal{B}$ capable of breaking the DDH assumption in $\mathbb{G}_1$ by using an adversary $\mathcal{A}$ that is able to win the chosen-plaintext with corruptions game with more than a negligible advantage.

We proof this though a series of hybrid games. Let game j be the game as defined in Definition 6, but where the first $j - 1$ components of the challenge query are replaced by random elements. Note that game 1 is identical to the original game and that it is not possible for any adversary to gain an advantage in game $n+1$. We are left to show that an adversary has at most a negligible advantage in distinguishing game j from game $j+1$.

Setup. The challenger $\mathcal{B}$ receives the bilinear group parameters and the DDH instance $(A = g_1^a, B = g_1^b, Z) \in (\mathbb{G}_1)^3$. It chooses the hash function H and the encryption keys usk_i. It sets encryption key $\mathsf{usk}_j = (A, \beta_j \xleftarrow{R} \mathcal{K}, \gamma_j \xleftarrow{R} \mathbb{Z}_p^*)$ and chooses the rest of the encryption keys according to the scheme. The public parameters and the encryption keys usk_i are given to the adversary.

Challenge. The adversary $\mathcal{A}$ submits an identifier id^* and two vectors $\boldsymbol{x}_0^*$, $\boldsymbol{x}_1^*$ to the challenger. The challenger chooses $b \xleftarrow{R} \{0,1\}$ and sets $g_1^{r_j} = B$. Additionally, it picks values $r_i \xleftarrow{R} \mathbb{Z}_p^*$ for $1 \le i \ne j \le n$. It gives the challenge

$$\mathsf{ct}_i = \begin{cases} \left(H(\mathsf{id}^*), g_1^{r_i}, R \xleftarrow{R} \mathbb{G}_1\right) & \text{if } i < j \\ \left(H(\mathsf{id}^*), B, Z^{\pi(\beta_i, x_{b,i}^*)} H(\mathsf{id}^*)^{\gamma_j}\right) & \text{if } i = j \\ \left(H(\mathsf{id}^*), g_1^{r_i}, g_1^{\alpha_i r_i \pi(\beta_i, x_{b,i}^*)} H(\mathsf{id}^*)^{\gamma_i}\right) & \text{if } i > j \end{cases}$$

for $1 \le i \le n$ to the adversary.

If the challenger is given $Z = g_1^{ab}$, then challenge ciphertext is identically distributed as the challenge ciphertext in game j and component j is a real encryption. If the challenger is given $Z \xleftarrow{R} \mathbb{G}_1$, then challenge ciphertext is identically distributed as the challenge ciphertext in game $j+1$ and component j is a random encryption.

Guess. The challenger outputs its guess that $Z = g_1^{ab}$ if the adversary guesses that it is playing game j, and outputs its guess that $Z \xleftarrow{R} \mathbb{G}_1$ if the adversary guesses that it is playing game $j+1$.

If the adversary has a non-negligible advantage in distinguishing between game j and game $j+1$, the challenger obtains a non-negligible advantage in solving the DDH problem in group $\mathbb{G}_1$.

References

1. Abdalla, M., Catalano, D., Dent, A.W., Malone-Lee, J., Neven, G., Smart, N.P.: Identity-based encryption gone wild. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 300–311. Springer, Heidelberg (2006). https://doi.org/10.1007/11787006_26
2. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 601–626. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_21
3. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13

4. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 563–594. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_19
5. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
6. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_29
7. Bösch, C., Hartel, P., Jonker, W., Peter, A.: A survey of provably secure searchable encryption. CSUR **47**(2), 18:1–18:51 (2014)
8. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_12
9. Chase, M., Meiklejohn, S., Zaverucha, G.: Algebraic MACs and keyed-verification anonymous credentials. In: CCS, pp. 1205–1216. ACM (2014)
10. Chenette, N., Lewi, K., Weis, S.A., Wu, D.J.: Practical order-revealing encryption with limited leakage. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 474–493. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_24
11. Conrad, S.H., LeClaire, R.J., O'Reilly, G.P., Uzunalioglu, H.: Critical national infrastructure reliability modeling and analysis. Bell Labs Tech. J. **11**(3), 57–71 (2006)
12. Dunn-Cavelty, M., Suter, M.: Public-private partnerships are no silver bullet: an expanded governance model for critical infrastructure protection. Int. J. Crit. Infrast. Prot. **2**(4), 179–187 (2009)
13. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. **156**(16), 3113–3121 (2008). Applications of Algebra to Cryptography
14. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS, pp. 40–49 (2013)
15. Goldwasser, S., et al.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_32
16. Gordon, S.D., Katz, J., Liu, F.H., Shi, E., Zhou, H.S.: Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774 (2013)
17. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
18. Lewi, K., Wu, D.J.: Order-revealing encryption: new constructions, applications, and lower bounds. In: CCS. ACM (2016)
19. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_31
20. Luiijf, E., Klaver, M.: On the sharing of cyber security information. In: Rice, M., Shenoi, S. (eds.) ICCIP 2015. IAICT, vol. 466, pp. 29–46. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26567-4_3

21. Luiijf, E., Nieuwenhuijs, A., Klaver, M., van Eeten, M., Cruz, E.: Empirical findings on critical infrastructure dependencies in Europe. In: Setola, R., Geretshuber, S. (eds.) CRITIS 2008. LNCS, vol. 5508, pp. 302–310. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03552-4_28
22. Miyaji, A., Nakabayashi, M., Takano, S.: Characterization of elliptic curve traces under FR-reduction. In: Won, D. (ed.) ICISC 2000. LNCS, vol. 2015, pp. 90–108. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45247-8_8
23. Moteff, J.D., Stevens, G.M.: Critical infrastructure information disclosure and homeland security (2002). http://www.dtic.mil/docs/citations/ADA467310
24. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. ACM **51**(2), 231–262 (2004)
25. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010)
26. Pandey, O., Rouselakis, Y.: Property preserving symmetric encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 375–391. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_23
27. President's Commission on Critical Infrastructure Protection: Critical foundations: Protecting America's infrastructures (1997). https://www.fas.org/sgp/library/pccip.pdf
28. Sahai, A., Waters, B.: Fuzzy Identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
29. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. ACM **27**(4), 701–717 (1980)
30. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_27
31. Shi, E., Chan, T.H., Rieffel, E.G., Chow, R., Song, D.: Privacypreserving aggregation of time-series data. In: NDSS. The Internet Society (2011). https://www.ndss-symposium.org/ndss2011/privacy-preserving-aggregation-of-time-series-data/
32. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18
33. Skopik, F., Settanni, G., Fiedler, R.: A problem shared is a problem halved: a survey on the dimensions of collective cyber defense through security information sharing. Comput. Secur. **60**, 154–176 (2016)
34. Smart, N.P.: The exact security of ECIES in the generic group model. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 73–84. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45325-3_8
35. Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic public key encryption with equality test. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 119–131. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11925-5_9